RWClustering

C++ Implementation of Rajaraman-Wong Clustering

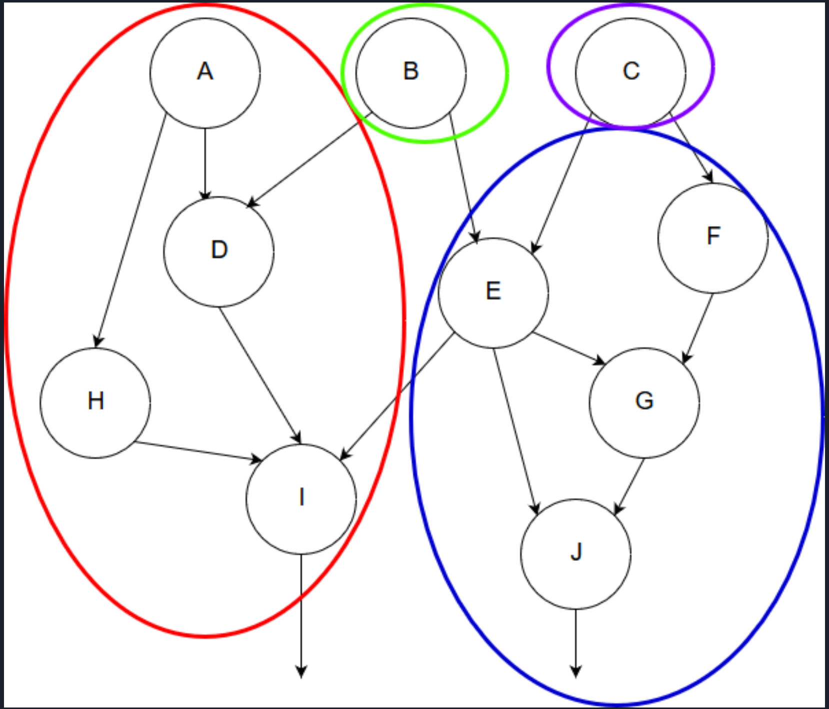
Akshay Nagendra <akshaynag@gatech.edu>
Paul Yates <paul.maxyat@gatech.edu>

April 26th, 2018

What is Clustering and Why Do We Do It?

- Clustering: Method of grouping gates together in a circuit to simplify complex circuit netlists

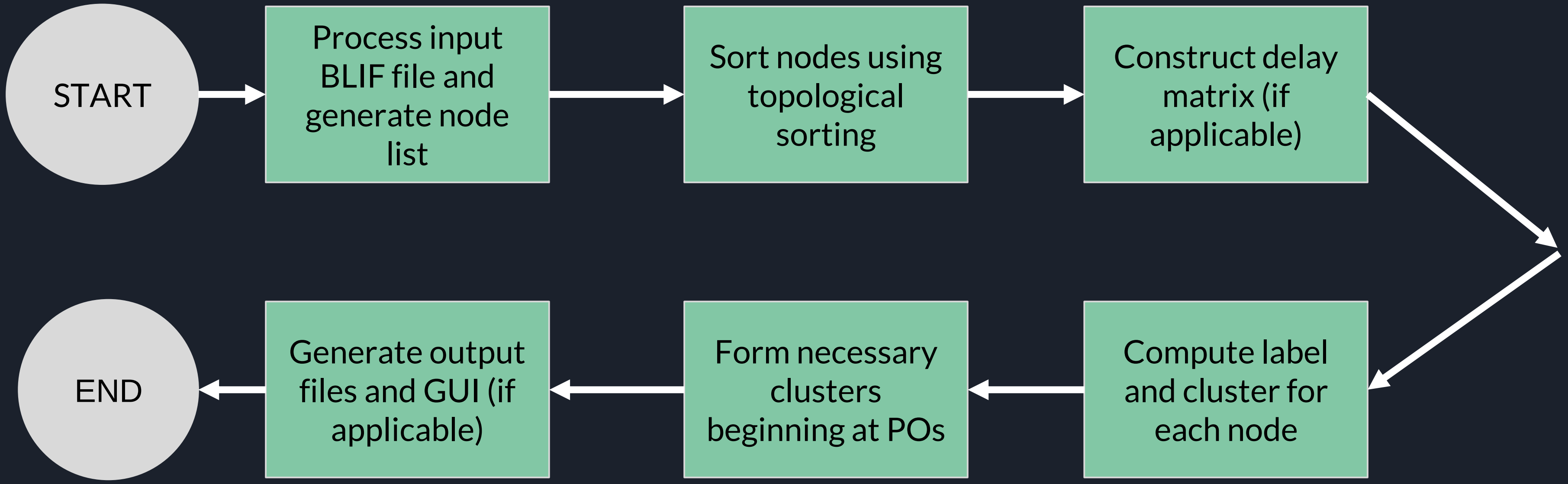
- Clustered netlists can be used as the input for partitioning and placement



Rajaraman-Wong Clustering

- Focuses on minimizing the critical path delay when forming clusters on a directed acyclic graph (DAG)
- Allows a gate to appear in multiple clusters (overlap)
  - Area cost
- Two Phases
  - Labelling: Compute a “label” per node that corresponds to longest time to that node when considering clustering
  - Clustering: Form certain node clusters by starting with POs and adding any necessary input clusters

Program Execution Flow



START

Process input BLIF file and generate node list

Sort nodes using topological sorting

Construct delay matrix (if applicable)

Compute label and cluster for each node

Form necessary clusters beginning at POs

Generate output files and GUI (if applicable)

END

Implementation: Data structures

- C++ and C++ STL libraries used to run all steps of the Rajaraman-Wong clustering algorithm
  - Object-oriented classes to store node and cluster information
  - Specialized traversal algorithms for topological sorting using visited flags, sparse matrices, C++ vectors and sets

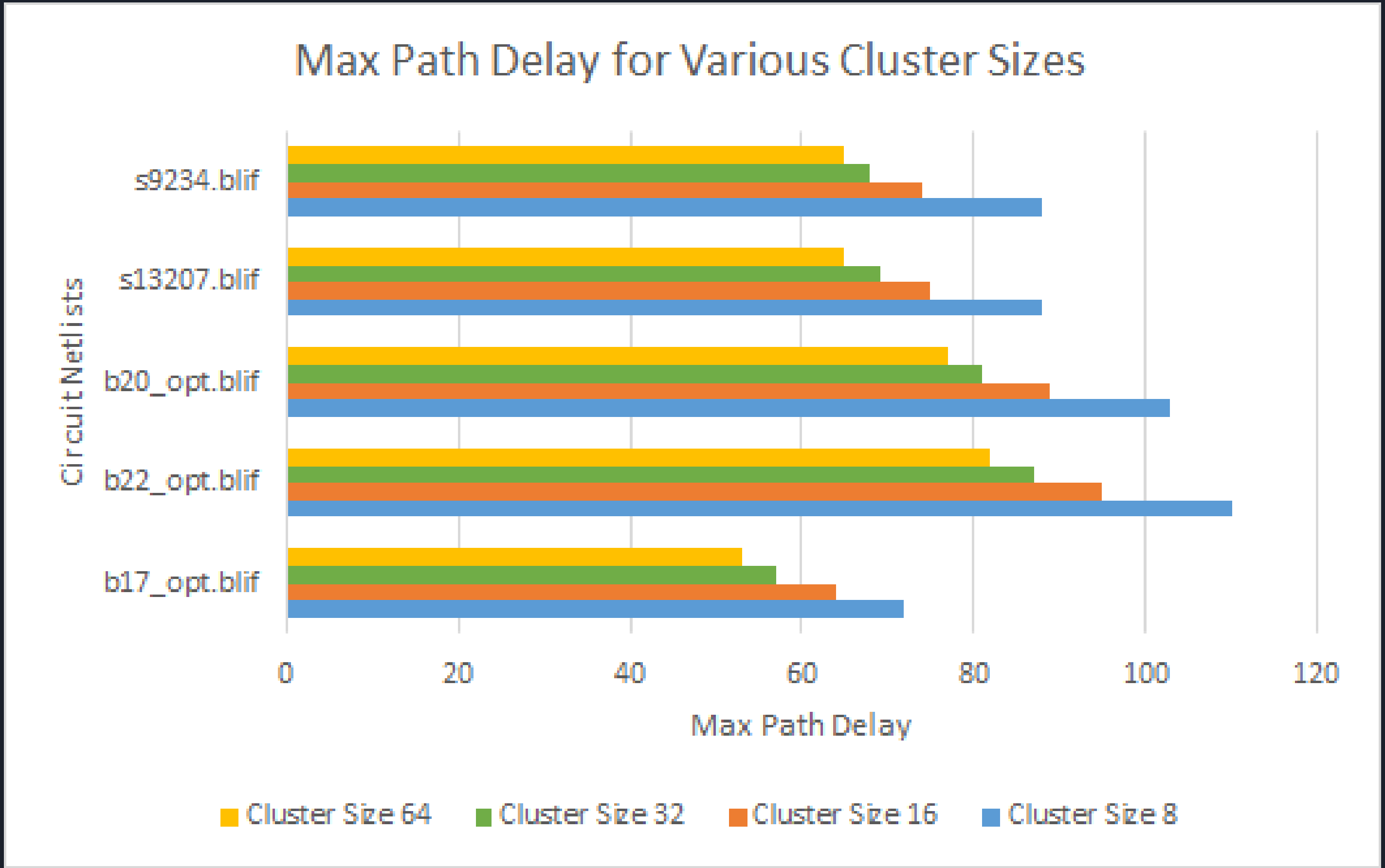
- GUI implemented using Python 2.7 Tkinter/Turtle graphics library and cshell script for running the C++ and the Python code

Results (Inter cluster delay = 3, Max cluster size = 8)

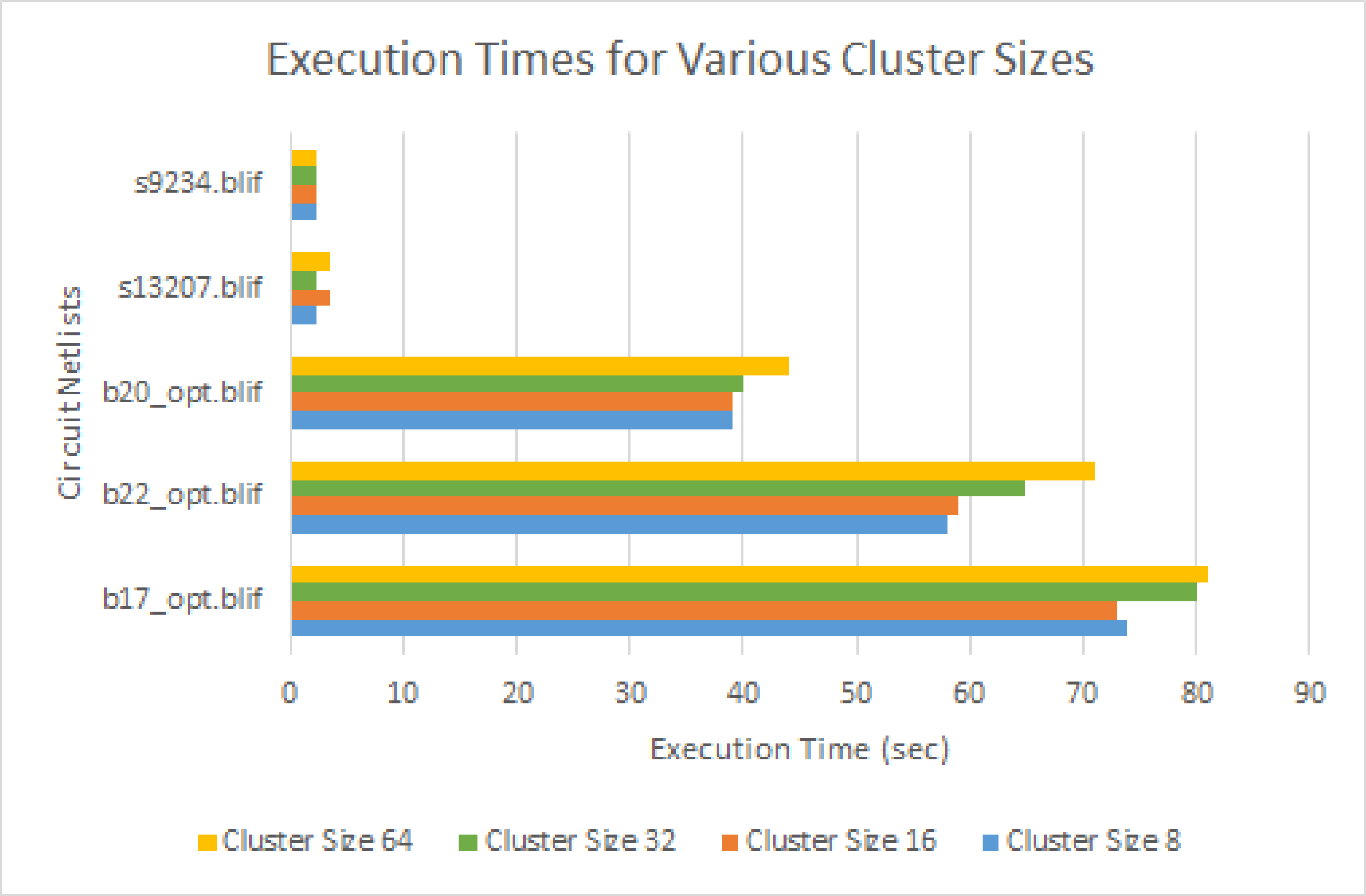
Circuit Netlist	Node Count	Cluster Count	Max Path Delay	Topolog. Sorting Time (sec)	Matrix Initialization (sec)	Labelling Time (sec)	Clustering Time (sec)	Total Execution Time (sec)
b17_opt.blif	25719	16147	72	0.003	27	5	9	74.003
b22_opt.blif	18789	11950	110	0.002	16	2	4	58.002
b20_opt.blif	12991	8197	103	0.001	11	1	2	39.001
s13207.blif	9396	2985	88	0.001	0.999	0.337	0.076	2.414
s9234.blif	6055	2090	88	0.00079	2	0.188	0.035	2.223

Note: These runs were executed on **ecelinsrvw.ece.gatech.edu**

Max Path Delay vs. Cluster Size Analysis



Execution Time vs. Cluster Size Analysis

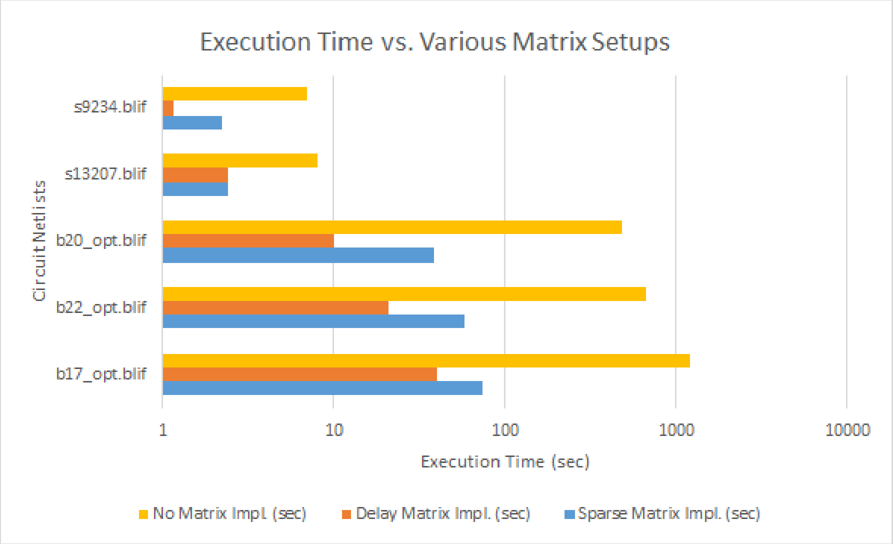


Performance Analysis

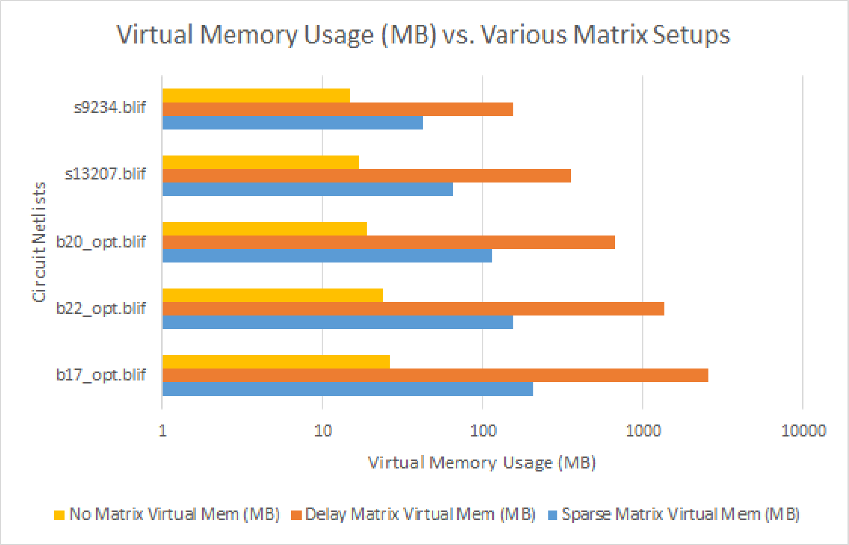
- Execution time and performance is not only a function of node count, but also the complexity of the circuit
  - Our implementation is more susceptible to higher degrees of complexity (*bX_opt.blif* circuits), compared to the simpler (*sX.blif* circuits)

- Additional features such as different matrix implementations help reduce the latency of the program

Matrix Implementation Analysis



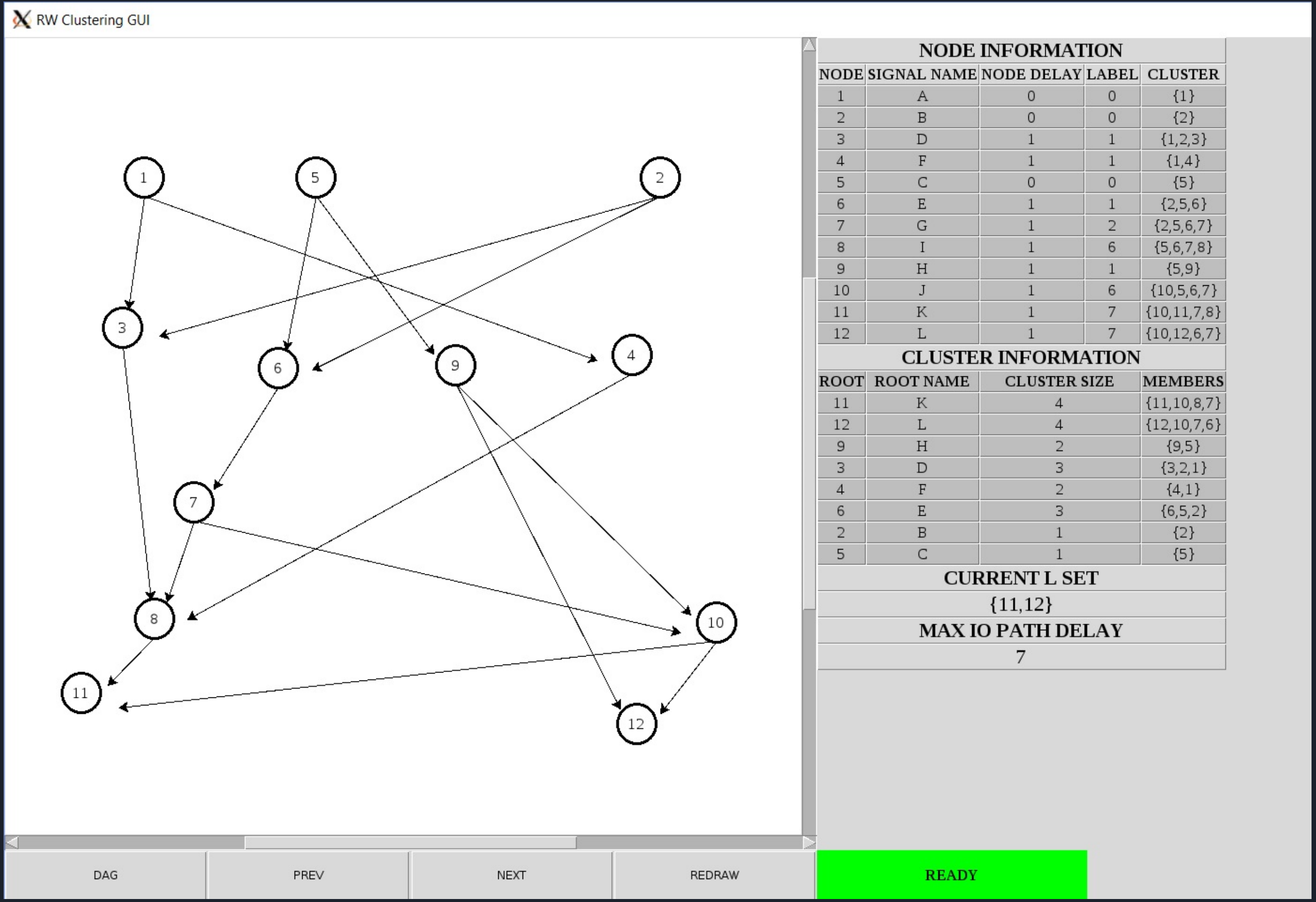
- Larger the matrix dimensions for the delay matrix, the better runtime performance can be achieved at a very high memory usage cost



Benefits of the Design

- *RWClustering* supports many features and extensions geared towards enabling a user to explore the clustering problem space
- Matrix configurations can be selected to prioritize specific goals (e.g. runtime, memory usage, etc.)
- Extensions include
  - Lawler labelling and clustering for comparison with Rajaraman-Wong
  - Experimental, overlap-avoidance clustering code to minimize area cost with Rajaraman-Wong
  - Python interactive GUI for sufficiently small circuits

GUI (*RWGUI*)



NODE INFORMATION

NODE	SIGNAL NAME	NODE DELAY	LABEL	CLUSTER
1	A	0	0	{1}
2	B	0	0	{2}
3	D	1	1	{1,2,3}
4	F	1	1	{1,4}
5	C	0	0	{5}
6	E	1	1	{2,5,6}
7	G	1	2	{2,5,6,7}
8	I	1	6	{5,6,7,8}
9	H	1	1	{5,9}
10	J	1	6	{10,5,6,7}
11	K	1	7	{10,11,7,8}
12	L	1	7	{10,12,6,7}

CLUSTER INFORMATION

ROOT	ROOT NAME	CLUSTER SIZE	MEMBERS
11	K	4	{11,10,8,7}
12	L	4	{12,10,7,6}
9	H	2	{9,5}
3	D	3	{3,2,1}
4	F	2	{4,1}
6	E	3	{6,5,2}
2	B	1	{2}
5	C	1	{5}

CURRENT L SET

{11,12}

MAX IO PATH DELAY

7

Future Work

- Further optimization of algorithms
  - E.g. optimize the “On-The-Fly”, No Matrix delay determination for better runtime performance
- Robust testing of the experimental code to see if it holds true for all acyclic digital circuit netlists
- Port *RWGUI* to a more sophisticated GUI platform such as OpenGL, Qt, or ElectronJS
  - Python Tkinter/Turtle was picked due to *ecelinsrvw.ece.gatech.edu* having support for this graphics library

Further Resources

- For more information about *RWClustering*, please consult the following:
    - *RWClustering/docs/RWGUIManual.doc*x Manual on how to operate the interactive GUI
    - *RWClustering/docs/FinalReport.docx*: Report on the *RWClustering* application with in-depth details about the algorithms used, features, extensions, and analyses
    - *RWClustering*/README.txt: Textfile containing instructions on how to run the project infrastructure